

## **BrainGoated - Project Planning Report**

**CMPT 276 - D200: Introduction to Software Engineering**

**Manjari Prasad**

**Alisha Maheebub Jesani**

**Gurpreet Kaur**

**Beyzanur Kuyuk**

**GitHub Repository : final-project-11-stars**

**<https://github.com/CMPT-276-SPRING-2025/final-project-11-stars.git>**

## Two Main APIs

Our two final APIs are The Trivia API and OpenAI API.

1. **The Trivia API** – This API provides a wide range of quiz questions across different topics and difficulty levels. It assists in creating a dynamic and engaging trivia experience by offering diverse question formats.

- a. Category and Difficulty Selection:

- i. *Description:* It allows users to select a specific trivia category (Science, History, Sports, etc.) and set a difficulty level (Easy, Medium, Hard) before fetching questions.
    - ii. *Benefit:* Users can personalize their quiz experience, making it engaging and suitable for their knowledge level. This feature also ensures that players don't get overwhelmed by difficulty or bored with overly simple questions.

- b. Fetching Text Choice Questions:

- i. *Description:* Retrieves multiple-choice trivia questions in text format, where the user selects one correct answer from four options.
    - ii. *Benefit:* Provides a structured and familiar quiz experience. Text-based questions load faster and are accessible to users.

- c. Fetching Image Choice Questions:

- i. *Description:* Retrieves trivia questions which include images as part of the question or answer choices
    - ii. *Benefit:* Enhances user engagement and caters to visual learners. For example, a geography quiz question could show images of different flags, asking the user to select the flag of a certain country. This makes the quiz more interactive and immersive.

2. **OpenAI API** – It enables AI-powered features such as generating personalised explanations. Additionally, it enhances the quiz experience by providing intelligent insights and interactive content.

a. AI- Generated Quiz Based on User Preference

- i. *Description:* Generates unique quiz questions based on user-selected themes, difficulty levels, or specific interests (eg., Create a quiz about space exploration at an advanced difficulty)
- ii. *Benefit:* Provides a tailored quiz experience that goes beyond standard trivia sets. Users can explore niche topics and enjoy a learning journey which matches their curiosity level.

b. Generating Fun Facts for Each Questions

- i. *Description:* After a user answers a question, the AI chatbot provides a fun fact related to the topic. For example, if a user asks a question about the Eiffel Tower, the API might generate: “Did you know the Eiffel Tower can grow up to 15 cm in hot weather due to metal expansion?”
- ii. *Benefit:* Adds an educational element by providing deeper context, hence making the quiz both informative and entertaining.

c. Answering User Follow-Up Questions

- i. *Description:* If a user wants to learn more about a trivia question, they could ask follow-up questions. The AI will provide a detailed, conversational response, keeping the user engaged and helping them further their understanding.
- ii. *Benefit:* Transforms the quiz from a simple Q&A into a dynamic learning experience. Users can deepen their understanding and satisfy their curiosity beyond just right or wrong answers.

**Changes Made:**

1. Removing the Teacher Page: The Trivia API does not support quiz creation or customization, making the teacher page redundant and inapplicable.
2. Remove the ‘Learn First’ Feature/Page: Instead of a separate learning page, we will incorporate AI- generated fun facts after each question to provide additional knowledge in an engaging way. This change ensures that the gamified nature of the quizzes remains intact while still incorporating educational content.

3. Add Image-Based Question Types: We have introduced a new feature which allows users to answer trivia questions with image-based choices, enhancing interactivity and catering to visual learners. This feature replaces the removed quiz creation feature for teachers, ensuring The Trivia API retains its required three features.

## Mid-Fidelity Prototype

Here is the link to our mid-fidelity prototype, created on Figma.

[BrainGoated Prototype](#)

## Chosen SDLC Model

We have chosen Agile-Kanban for the following reasons:

- It offers *continuous workflow and flexibility* as it allows teams to prioritize and adjust work dynamically, which leads to faster delivery and improved responsiveness to change
- The kanban board provides a *clear visual representation* of tasks in progress, which helps identify blockers and improve efficiency
- The *risk of project delays can be minimized* as new priorities can be added without waiting for a new sprint cycle
- It encourages *cross-functional collaboration* by allowing different team members to assist where needed as opposed to being locked into specific sprint tasks

## Work Breakdown Structure (WBS)

Tasks below are listed in order of importance and dependencies:

### 1) Project Setup

### **1.1) Set up project board (Kanban)**

- Create Labels for tasks (API, Frontend, etc.)
- Set Milestones and sprints
- Create Issues

### **1.2) Initialize Documentation**

- Initialize README
- Document API references

## **2) Frontend Development (React and Figma) - Dependent on 1**

### **2.1) Design interactive prototype**

- Finalize website fonts, colours, layout

### **2.2) Set up React Project**

- Set up React using CRA
- Install necessary dependencies (react-router)
- Set up folder structure (components, pages, context, styles, assets)

### **2.3) Implement base structure of website**

- Build:
  - Homepage
  - Quiz Categories section
  - Quiz Page
- Set up navigation system (React Router)
- Build reusable components
  - Buttons
  - Quiz Category
  - Quiz Question (Text-based and Image-based)
  - Pop-ups

### **2.4) Implement State Management**

- Set up state for Category and Difficulty selection using useState
- Set up state for Question Type (Image-based vs Text-based)

- Manage API data with useState() - storing fetched questions, AI generated material, etc.
- Implement state for user quiz interaction (tracking current question, storing selected answer, tracking number of correct answers, ensuring quiz progress reset when starting new quiz)
- Implement state for timer and score tracking

## **2.5) Create Mock Data for State Management and UI logic testing**

- Test category and difficulty selection updating properly
- Test component interactions
- Test page navigation and data persistence (Ensure React Router preserves state)
- Simulate API responses to test:
  - Quiz state updating properly
  - Answers being recorded and score being updated correctly
  - Incorrect/Correct pop-ups working/updating correctly (displaying of mock fun-facts and AI chatbot)

## **3) The Trivia API Integration - Dependent on 2**

### **3.1) Implement Text Based vs Image Based Quizzes**

- Construct API request based on user selection of Text or Image-Based quizzes

### **3.2) Implement Category and Difficulty selection**

- Construct API request based on user selection of category and difficulty
- Handle API response and store questions in state

### **3.3) Connect to UI in React and test features**

- Display fetched questions in the quiz UI
- Ensure previously tested state management works properly after API integration
- Test API responses and handle errors

## **4) OpenAI API Integration - Dependent on 2 and 3**

#### **4.1) Implement AI-Generated Fun-Facts/Informative Message After Each Question**

- Connect to OpenAI API
- Create prompt, requesting fun-facts based on fetched questions from the Trivia API
- Store the AI generated response
- Display the generated fun-facts within the incorrect/correct pop-ups after each question

#### **4.2) Implement AI tutor chatbot (Bud-E) for answering user follow up questions**

- Capture user queries and send to OpenAI API
- Display AI responses dynamically without needing a page refresh
- Add user interface for conversation history

#### **4.3) Implement AI-Generated quiz based on user preference**

- Send user selected category and topic to OpenAI API with a prompt for generating trivia quiz questions
- Fetch quiz questions
- Display AI-generated quiz UI dynamically without having to refresh page

#### **4.4) Connect to UI in React and test features**

- Ensure state management functions properly
- Test API responses and handle errors

### **5) User Experience Enhancements - Dependent on 2**

#### **5.1) Develop Timer and Score tracking**

- Implement countdown timer
- Calculate and display user score

### **6) UI Styling and Enhancements using CSS**

#### **6.1) Homepage styling**

- Design introduction section with animations/motion effects

- Add BrainGoated logo/moscot
- Design category selection section and add category icons

## **6.2) Popup styling (Start Quiz)**

- Design interactive popup UI
- Add motion effects such as Fade-In

## **6.3) Quiz UI styling**

- Add loading message while quiz questions are being fetched
- Style and improve Quiz layout

## **6.4) Correct/Incorrect Feedback Popup styling**

- Style interactive popup UI
- Style correct choice and fun-fact displays
- Design and style AI chatbot tutor (Bud-e)

## **6.5) Score Display Popup styling**

- Design interactive popup UI for displaying score
- Add motion effects

# **7) Testing and Deployment - Dependent on all of the above**

## **7.1) API and UI Testing**

- Validate The Trivia API and OpenAI API responses
- Test UI components for correct rendering and interaction
- Identify and fix bugs

## **7.2) Final Documentation and Deployment**

- Implement CI/CD Pipeline
- Deploy website
- Final Documentation and README update
- Include step-by-step instructions for the project to be deployed locally (including instructions for installing dependencies)
- Include links to all project website, reports and videos
- Submit AI disclosure forms



## Project Schedule

Milestones:	Internal	External
M0 - Project Setup	Mar 7	Mar 8
M1 - Front-end Development - Base Structure and UI logic	Mar 11	Mar 12
M2 - Front-end Development - State Management	Mar 14	Mar 15
M3 - Front-end Development - UI logic and State Management Testing	Mar 15	Mar 16
M4 - The Trivia API Integration	Mar 18	Mar 19
M5 - Test The Trivia API Integration	Mar 19	Mar 20
M6 - OpenAI API Integration	Mar 23	Mar 24
M7 - Test OpenAI API Integration	Mar 24	Mar 26
M8 - User Experience Enhancements	Mar 26	Mar 28
M9 - UI Styling and Enhancements: Homepage, Popup, Quiz UI	Mar 30	Mar 31
M10 - UI Styling and Enhancements: Correct/Incorrect and Score popup	April 2	April 3
M11 - Testing and Deployment: API and UI Extensive Testing	April 3	April 4
M12 - Testing and Deployment: Implement CI/CD and Deploy	April 4	April 5
M13 - Testing and Deployment: Finalize Documentation	April 6	April 7

## Risk Assessment and Mitigation Strategies

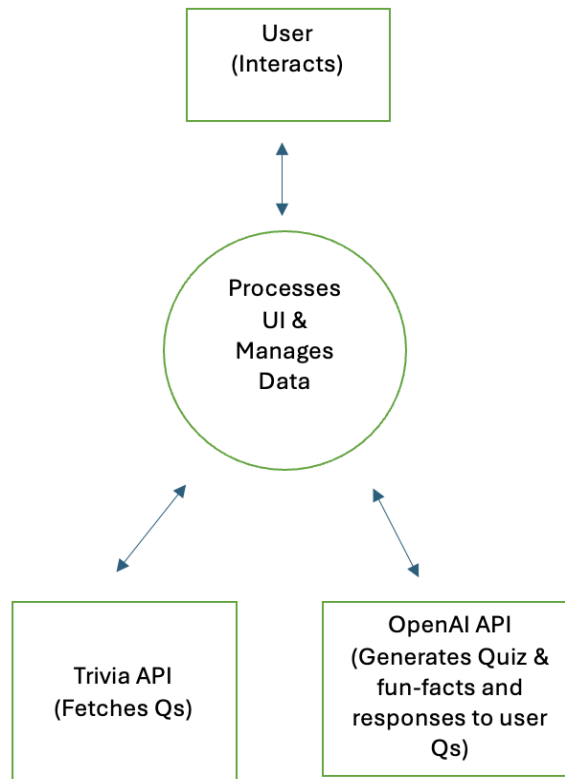
- **Low Risk:**
  - OpenAI Chatbot repeating responses

- *Mitigation:* We will adjust the prompt engineering in order to encourage more diverse AI-generated responses.
- Slow API Response time
  - *Mitigation:* We will implement a loading animation to improve user experience during inevitable delays.
- Trivia API returning irrelevant questions
  - *Mitigation:* We will allow users to report irrelevant questions to improve selection over time.
- Small Lag in AI Chatbot Response
  - *Mitigation:* We will optimize API requests by limiting unnecessary calls.
- Trivia API formatting issues
  - *Mitigation:* We will create a fallback mechanism that reformats or excludes broken questions.
- **Medium Risk:**
  - The Trivia API Questions not being age-appropriate
    - *Mitigation:* We will provide toggles for restricting categories and filtering out certain topics.
  - AI Chatbot Giving Overly Complicated Explanation:
    - *Mitigation:* Instead of full explanations, the chatbot will guide the users step by step with questions.
  - The Trivia API Not Updating Question Bank Frequently
    - *Mitigation:* We will use OpenTrivia API as a backup when The Trivia API provides outdated content.
  - API Rate Limits Reached
    - *Mitigation:* We will use multiple API keys and switch between them dynamically in order to bypass rate limits.

- OpenAI API Cost Scaling Issues
  - *Mitigation:* We will reduce wordiness in prompts sent to OpenAI to decrease API token usage.
- **High Risk:**
  - The Trivia API Goes Down
    - *Mitigation:* We will use the OpenTrivia API as a backup. In case there's an issue with OpenTrivia as well, we would store a local database of commonly used quiz questions.
  - OpenAI API Stops Responding:
    - *Mitigation:* We would provide pre-written explanations for common queries.
  - AI Chatbot Generating Inappropriate Responses:
    - *Mitigation:* We will restrict AI from generating responses on sensitive and inappropriate topics using prompt engineering.
  - Excessive API Costs if Usage Spikes:
    - *Mitigation:* We would optimize chatbot responses by batching multiple queries within a single API call.
  - AI Chatbot Overloading and Crashing the System
    - *Mitigation:* We would restrict how frequently users can send queries to the chatbot in order to prevent overload.

## Data Flow Diagrams:

### Level 0:



### Overview:

The **Level 0 DFD** represents the high-level data flow within the **BrainGoated System**, focusing on interactions between users, APIs, and the system. It outlines the movement of quiz data, user interactions, and AI-generated responses.

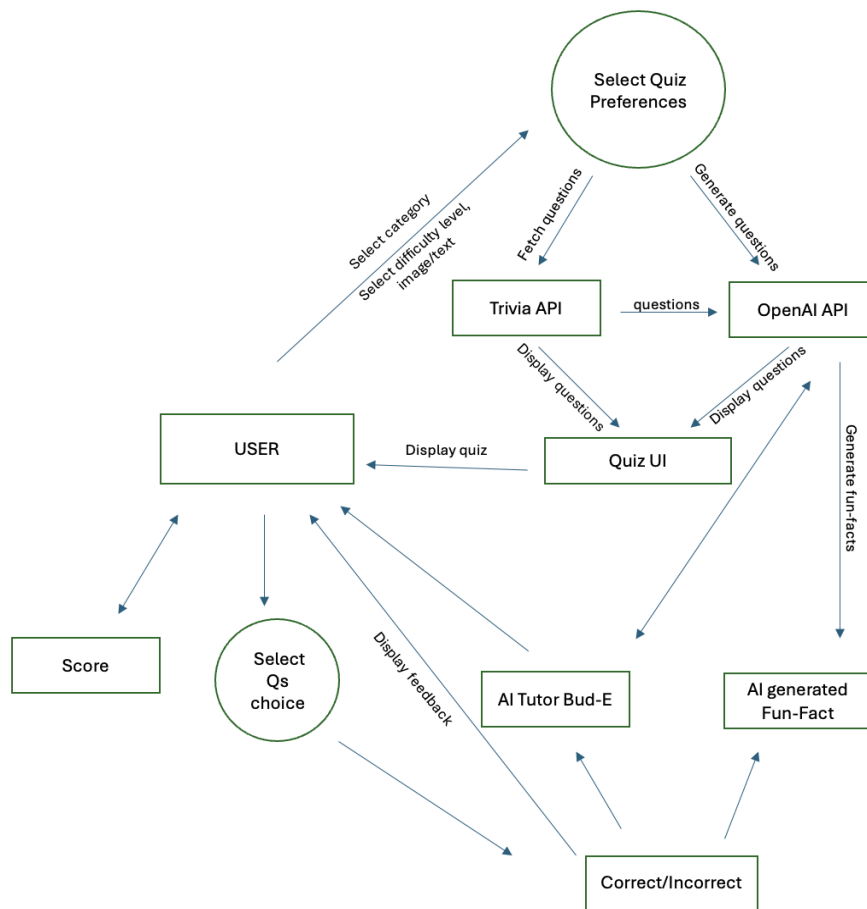
### **Entities & Processes:**

1. **User** – Provides inputs like quiz category, difficulty, and AI-generated quiz requests.
2. **BrainGoated System** – Manages user requests, fetches quiz data, and provides interactive responses.
3. **The Trivia API** – Supplies quiz questions (both text and image-based).
4. **OpenAI API** – Generates AI-based quizzes, fun facts, and follow-up explanations.

### **Data Flow:**

1. **User Inputs Preferences** → BrainGoated System processes request.
2. **Trivia API Fetches Questions** → System stores and displays them.
3. **OpenAI API Generates Questions, Fun Facts & Explanations** → System presents them to the user.
4. **User Answers Question** → System evaluates and updates scores.

### **Level 1:**



## Overview:

The **Level 1 DFD** expands on Level 0, detailing how quiz questions are retrieved, processed, and displayed dynamically.

## User Interaction:

### 1. User Selects Quiz Options:

- Picks **category** (e.g., Science, History).

- Chooses **difficulty** (Easy, Medium, Hard).
  - Selects **quiz type** (Text/Image-based).
  - 2. **User Requests AI-Generated Quiz:**
    - System sends request to **OpenAI API** for a quiz based on preferences.
  - 3. **User Answers Questions:**
    - System processes **correct/incorrect answers**.
    - AI generates **fun facts** after each question.
  - 4. **User Asks Follow-up Questions:**
    - OpenAI API provides **detailed explanations**.
- 

## Data Processing & API Requests:

1. **Trivia API Request:**
    - System sends request for quiz questions.
    - Trivia API returns questions & choices.
    - System formats data and updates UI.
  2. **OpenAI API Request:**
    - System sends user preferences for **AI-generated quizzes**.
    - OpenAI API returns generated questions and explanations.
- 

## Workflows

### 1. User Takes a Quiz (Trivia API)

1. User selects the **category and difficulty**.
2. BrainGoated System requests **questions from The Trivia API**.
3. API returns **formatted questions**.
4. User answers a question.
5. System evaluates response (correct/incorrect).
6. System updates **score**.

### 2. AI-Generated Fun Facts (OpenAI API)

1. User submits an answer.
2. BrainGoated System sends the **question topic to OpenAI API**.

3. API returns **a fun fact**.
4. Fun fact is displayed in the quiz interface.

### 3. AI-Generated Quiz (OpenAI API)

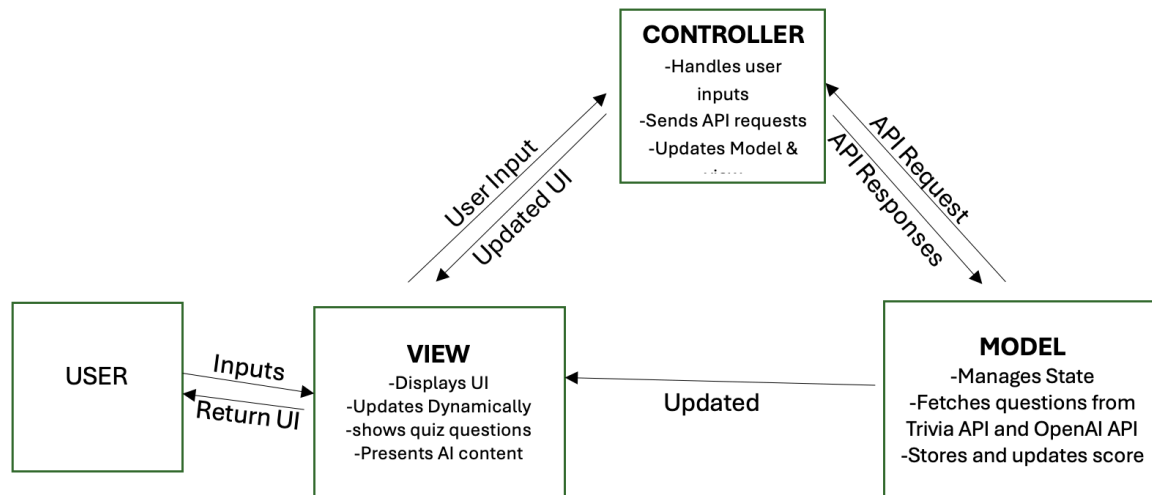
1. User selects **custom quiz settings**.
2. BrainGoated System **requests AI-generated questions**.
3. OpenAI API returns **a new quiz**.
4. Questions are displayed dynamically.

### 4. AI Chatbot Follow-up Questions

1. User asks for **more details on a question**.
2. BrainGoated System **sends query to OpenAI API**.
3. AI generates a **detailed explanation**.
4. System displays the **answer**.



## MVC Model Diagram



The BrainGoated Prototype follows the **Model-View-Controller (MVC)** architectural pattern to structure the application efficiently. This design pattern ensures separation of concerns, making the application more scalable and maintainable.

### Component Breakdown

#### 1. Model (M)

The **Model** is responsible for handling all data-related operations, including fetching quiz questions from APIs, managing AI-generated responses, and maintaining application state. It communicates with the API's to make the interface smooth and seamless.

- **Key Responsibilities:**

- Fetch trivia questions from The Trivia API based on user-selected categories and difficulty levels.
- Retrieve AI-generated quiz questions and explanations from OpenAI API.

- Process and store user-selected answers and scores in the application state.
- Manage application state for tracking quiz progress.
- **Data Flow:**
  - Retrieves questions from The Trivia API and OpenAI API.
  - Stores user selections, quiz history, and AI-generated explanations in frontend state.
  - Updates the View with relevant information upon changes.

## 2. View (V)

The **View** is responsible for managing all UI elements and ensuring an intuitive user experience. It presents the fetched data, updates dynamically based on user interaction, and enhances accessibility with visual and interactive elements.

- **Key Responsibilities:**
  - Displays trivia questions and answer choices.
  - Presents AI-generated explanations for questions.
  - Manages user interaction elements, including buttons, timers, and score tracking.
  - Provides animated transitions and responsive UI design.
- **Components:**
  - **Quiz Interface:** Displays questions and choices for users.
  - **Learning Module:** Shows AI-generated facts and explanations.
  - **User Progress & Score Display:** Updates scores and tracks user performance.
  - **Navigation System:** Enables smooth movement between sections.

## 3. Controller (C)

The **Controller** acts as the intermediary between the Model and View, handling user inputs, making API requests, and updating both the UI and data accordingly.

- **Key Responsibilities:**
  1. Processes user inputs (e.g., selecting quiz categories, submitting answers, asking follow-up questions to the AI chatbot).

2. Sends requests to The Trivia API and OpenAI API directly from the frontend.
3. Updates the Model with received data and ensures state management consistency.
4. Controls application logic, such as starting/stopping quizzes and updating scores.
5. Manages API failure handling and fallback mechanisms.

- **Workflow:**

1. User interacts with UI elements (View).
2. Controller captures the interaction and sends requests to the Model.
3. Model fetches data from APIs and updates stored state in the frontend.
4. View receives updated data from the Model and re-renders UI dynamically.

## **High-Level Overview of Application Structure**

- **Frontend (React + CSS) – View (V)**
  - Manages UI layout, animations, and component interactions.
  - Uses React state management to display quiz questions dynamically.
- **API Calls & State Management – Model (M)**
  - Fetches data from The Trivia API and OpenAI API.
  - Stores user progress and manages API responses in local state.
- **Logic & Event Handling – Controller (C)**
  - Handles API request calls from the frontend.
  - Processes user interactions.
  - Updates the UI accordingly.

## **Appendix**

### **A. Group Members and Contributions**

This section outlines the contributions of each group member to the Project and the Planning Report.

#### **Manjari**

##### Assigned tasks for the project:

- The Trivia API Integration
- OpenAI API Integration
- User Experience and Enhancements
- Deployment (CI/CD)

##### Contributions to Milestone 1:

- Finalized API selection and its features, along with the SDLC Model, providing detailed descriptions.
- Conducted risk assessment, identifying potential challenges and mitigation strategies.
- Created PowerPoint Presentation for the video presentation.

#### **Beyzanur**

##### Assigned tasks for the project:

- Front-end Development - Implementing state management
- OpenAI API Integration
- User Experience and Enhancements
- Deployment (CI/CD)

##### Contributions to Milestone 1:

- Developed the Mid-Fidelity Prototype, outlining the user interface and interactions.

- Created the Work Breakdown Structure (WBS) to organize project tasks and created GitHub issues

## **Alisha**

### Assigned tasks for the project:

- Front-end Development - Implement base structure and navigation of the web app.
- UI Styling and Enhancements using CSS
- Create Mock Data for State Management and UI logic testing
- Final Testing (API and UI Testing)

### Contributions to Milestone 1:

- Designed the MVC Model Diagram, outlining the system's architecture.
- Created and edited the video presentation

## **Gurpreet**

### Assigned tasks for the project:

- Front-end Development - Implement base structure and navigation of the web app.
- UI Styling and Enhancements using CSS
- Create Mock Data for State Management and UI logic testing
- Final Testing (API and UI Testing)

### Contributions to Milestone 1:

- Created the Data Flow Diagram (DFD) to illustrate data movement between APIs and the application.
- Created and edited the video presentation

## **B. Changelog – Revisions Since Proposal**

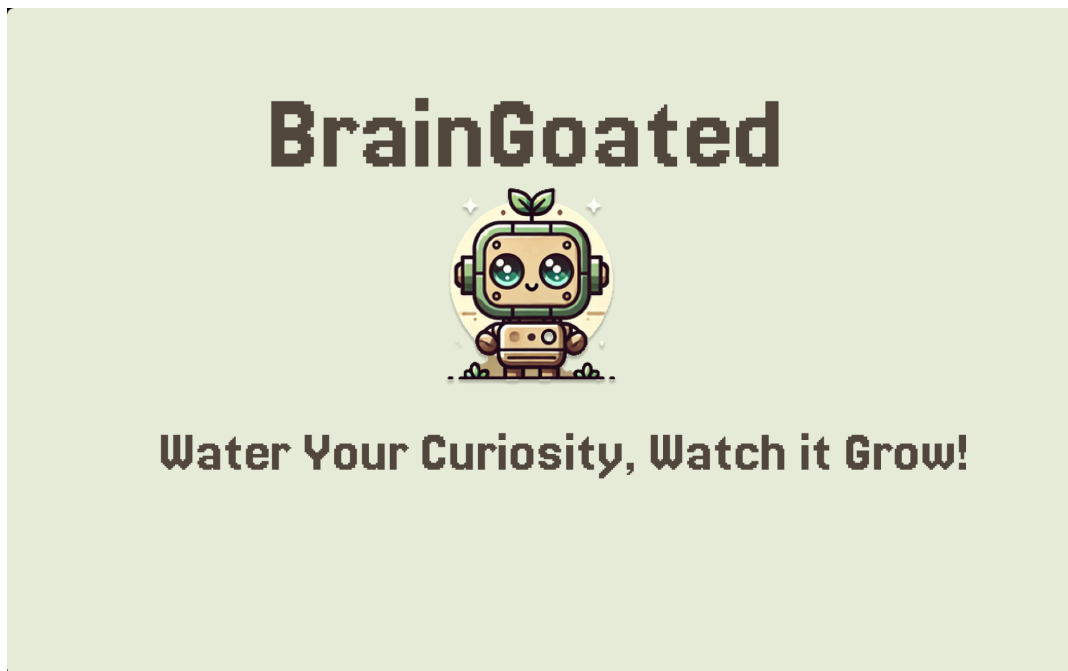
The following table documents significant revisions made after the initial project proposal.

Date	Change	Reason for Revision
February 21, 2025	Removed the Teacher Page	The Trivia API does not support quiz creation or customisation.
February 21, 2025	Added Image-Based Question Types	Enhances user experience by incorporating visual elements.  (Replaces The Trivia API quiz creation feature.)
February 23, 2025	Removed the 'Learn First' Page	To maintain the quiz's gamified nature without disrupting engagement.  (Replaced with AI-generated fun facts integrated after each question.)

## C: Additional Diagrams

- a. Mid-Fidelity Prototype Screens:

Homepage:



Quiz categories with options to choose difficulty, category, and type of questions:



Page for text-based quiz:

**Timer:**

**Score:**

**Q1: Which city is the capital city of Turkey?**

**A) Istanbul**

**B) Izmir**

**C) Ankara**

**D) Antalya**





**Exit**

Page for image-based quiz:

**Timer:**

**Score:**

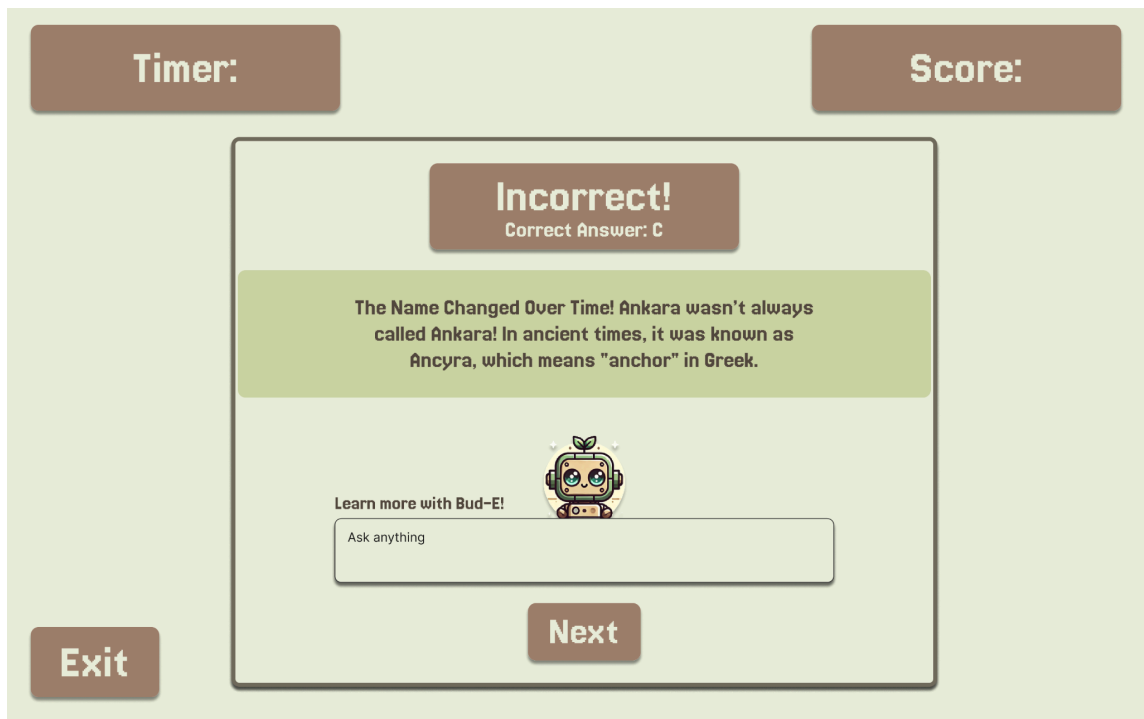
**Q1: Which city is the capital city of Turkey?**



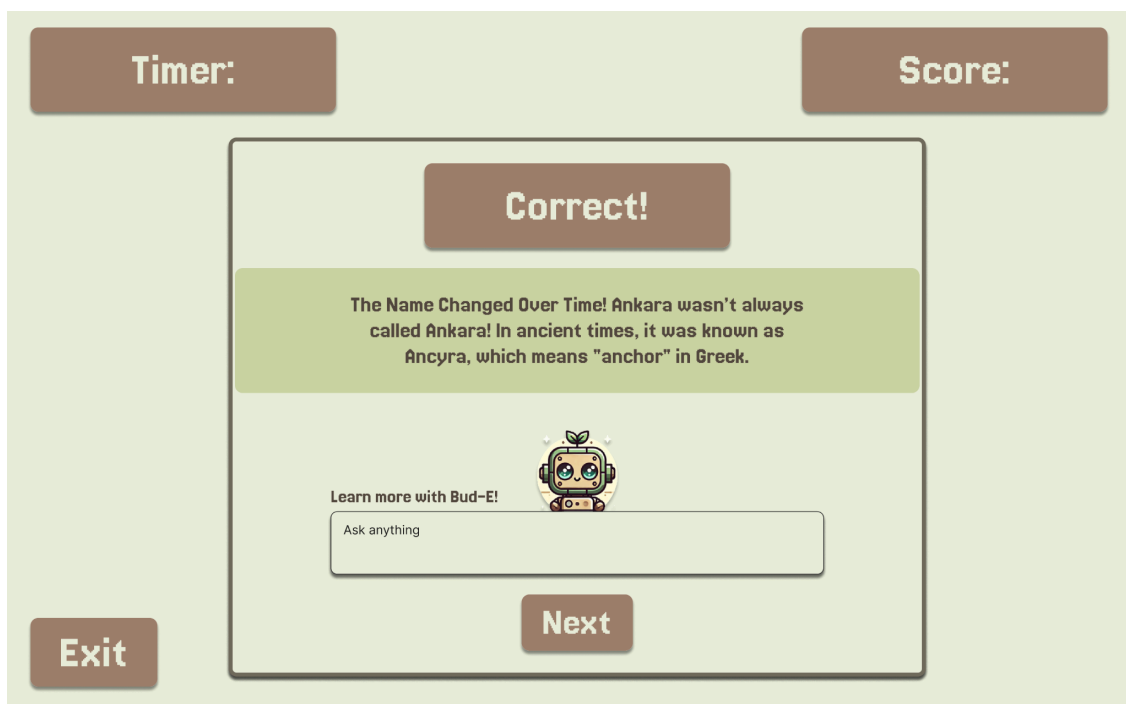
**Exit**



Popup for incorrect answer:



Popup for correct answer:



Last Page with words of motivation:



b. Project Schedule:

**Project Schedule**

Milestones:	Internal	External
<b>M0</b> - Project Setup	Mar 7	Mar 8
<b>M1</b> - Front-end Development - Base Structure and UI logic	Mar 11	Mar 12
<b>M2</b> - Front-end Development - State Management	Mar 14	Mar 15
<b>M3</b> - Front-end Development - UI logic and State Management Testing	Mar 15	Mar 16
<b>M4</b> - The Trivia API Integration	Mar 18	Mar 19
<b>M5</b> - Test The Trivia API Integration	Mar 19	Mar 20
<b>M6</b> - OpenAI API Integration	Mar 23	Mar 24
<b>M7</b> - Test OpenAI API Integration	Mar 24	Mar 26
<b>M8</b> - User Experience Enhancements	Mar 26	Mar 28
<b>M9</b> - UI Styling and Enhancements: Homepage, Popup, Quiz UI	Mar 30	Mar 31
<b>M10</b> - UI Styling and Enhancements: Correct/Incorrect and Score popup	April 2	April 3
<b>M11</b> - Testing and Deployment: API and UI Extensive Testing	April 3	April 4
<b>M12</b> - Testing and Deployment: Implement CI/CD and Deploy	April 4	April 5
<b>M13</b> - Testing and Deployment: Finalize Documentation	April 6	April 7